What’s New in Sun WorkShop 6

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
U.S.A. 650-960-1300

Part No. 806-3565-10
May 2000, Revision A

Send comments about this document to: `docfeedback@sun.com`

Please
Recycle

Adobe PostScript

Important Note on New Product Names

As part of Sun's new developer product strategy, we have changed the names of our development tools from Sun WorkShop™ to Forte™ Developer products. The products, as you can see, are the same high-quality products you have come to expect from Sun; the only thing that has changed is the name.

We believe that the Forte™ name blends the traditional quality and focus of Sun's core programming tools with the multi-platform, business application deployment focus of the Forte tools, such as Forte Fusion™ and Forte™ for Java™. The new Forte organization delivers a complete array of tools for end-to-end application development and deployment.

For users of the Sun WorkShop tools, the following is a simple mapping of the old product names in WorkShop 5.0 to the new names in Forte Developer 6.

Old Product Name	**New Product Name**
Sun Visual WorkShop™ C++	Forte™ C++ Enterprise Edition 6
Sun Visual WorkShop™ C++ Personal Edition	Forte™ C++ Personal Edition 6
Sun Performance WorkShop™ Fortran	Forte™ for High Performance Computing 6
Sun Performance WorkShop™ Fortran Personal Edition	Forte™ Fortran Desktop Edition 6
Sun WorkShop Professional™ C	Forte™ C 6
Sun WorkShop™ University Edition	Forte™ Developer University Edition 6

In addition to the name changes, there have been major changes to two of the products.

- Forte for High Performance Computing contains all the tools formerly found in Sun Performance WorkShop Fortran and now includes the C++ compiler, so High Performance Computing users need to purchase only one product for all their development needs.
- Forte Fortran Desktop Edition is identical to the former Sun Performance WorkShop Personal Edition, except that the Fortran compilers in that product no longer support the creation of automatically parallelized or explicit, directive-based parallel code. This capability is still supported in the Fortran compilers in Forte for High Performance Computing.

We appreciate your continued use of our development products and hope that we can continue to fulfill your needs into the future.

Contents

Tables

Preface

What's New in Sun WorkShop™ 6 describes the new features of the Sun WorkShop 6 compilers and tools.

Multiplatform Release

This Sun WorkShop release supports versions 2.6, 7, and 8 of the Solaris™ *SPARC™ Platform Edition* and Solaris *Intel Platform Edition* Operating Environments.

The Solaris operating environment runs on two types of hardware, or platforms—SPARC and IA. The Solaris environment also runs on both 64-bit and 32-bit address spaces. The information in this document pertains to both platforms and address spaces unless called out in a special chapter, section, note, bullet, figure, table, example, or code example.

Note – The terms "IA" and "x86" refer to the Intel 8086 family of microprocessor chips, including the Pentium, Pentium Pro, and Pentium II processors and compatible microprocessor chips made by AMD and Cyrix. In this document, the term "x86" refers to the overall platform architecture, whereas "*Intel Platform Edition*" appears in the product name.

Access to Sun WorkShop Development Tools

Because Sun WorkShop product components and man pages do not install into the standard `/usr/bin/` and `/usr/share/man` directories, you must change your `PATH` and `MANPATH` environment variables to enable access to Sun WorkShop compilers and tools.

To determine if you need to set your `PATH` environment variable:

1. **Display the current value of the `PATH` variable by typing:**

```
% echo $PATH
```

2. **Review the output for a string of paths containing `/opt/SUNWspro/bin/`.**

   If you find the paths, your `PATH` variable is already set to access Sun WorkShop development tools. If you do not find the paths, set your `PATH` environment variable by following the instructions in this section.

To determine if you need to set your `MANPATH` environment variable:

1. **Request the `workshop` man page by typing:**

```
% man workshop
```

2. **Review the output, if any.**

   If the `workshop`(1) man page cannot be found or if the man page displayed is not for the current version of the software installed, follow the instructions in this section for setting your `MANPATH` environment variable.

Note – The information in this section assumes that your Sun WorkShop 6 products were installed in the `/opt` directory. Contact your system administrator if your Sun WorkShop software is not installed in `/opt`.

The `PATH` and `MANPATH` variables should be set in your home `.cshrc` file if you are using the C shell or in your home `.profile` file if you are using the Bourne or Korn shells:

- To use Sun WorkShop commands, add the following to your `PATH` variable:

  `/opt/SUNWspro/bin`

- To access Sun WorkShop man pages with the `man` command, add the following to your `MANPATH` variable:

  ```
  /opt/SUNWspro/man
  ```

For more information about the `PATH` variable, see the `csh`(1), `sh`(1), and `ksh`(1) man pages. For more information about the `MANPATH` variable, see the `man`(1) man page. For more information about setting your `PATH` and `MANPATH` variables to access this release, see the *Sun WorkShop 6 Installation Guide* or your system administrator.

Typographic Conventions

TABLE P-1 shows the typographic conventions that are used in Sun WorkShop documentation.

TABLE P-1 Typographic Conventions

Typeface	Meaning	Examples
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail`.

`AaBbCc123`	What you type, when contrasted with on-screen computer output	`%` **`su`**
		`Password:`

AaBbCc123	Book titles, new words or terms, words to be emphasized	Read Chapter 6 in the *User's Guide.* These are called *class* options. You *must* be superuser to do this.
AaBbCc123	Command-line placeholder text; replace with a real name or value	To delete a file, type `rm` *filename*.

Shell Prompts

TABLE P-2 shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P-2 Shell Prompts

Shell	Prompt
C shell	%
Bourne shell and Korn shell	$
C shell, Bourne shell, and Korn shell superuser	#

Related Documentation

You can access documentation related to the subject matter of this book in the following ways:

- **Through the Internet at the `docs.sun.com`sm Web site.** You can search for a specific book title or you can browse by subject, document collection, or product at the following Web site:

  `http://docs.sun.com`

- **Through the installed Sun WorkShop products on your local system or network.** Sun WorkShop 6 HTML documents (manuals, online help, man pages, component readme files, and release notes) are available with your installed Sun WorkShop 6 products. To access the HTML documentation, do one of the following:
  - In any Sun WorkShop or Sun WorkShop™ TeamWare window, choose Help ➤ About Documentation.
  - In your Netscape™ Communicator 4.0 or compatible version browser, open the following file:

    `/opt/SUNWspro/docs/index.html`

  (Contact your system administrator if your Sun WorkShop software is not installed in the `/opt` directory.) Your browser displays an index of Sun WorkShop 6 HTML documents. To open a document in the index, click the document's title.

TABLE P-3 lists related Sun WorkShop 6 manuals by document collection.

TABLE P-3 Related Sun WorkShop 6 Documentation by Document Collection

Document Collection	**Document Title**	**Description**
Forte™ Developer 6 / Sun WorkShop 6 Release Documents	*About Sun WorkShop 6 Documentation*	Describes the documentation available with this Sun WorkShop release and how to access it.
	What's New in Sun WorkShop 6	Provides information about the new features in the current and previous release of Sun WorkShop.
	Sun WorkShop 6 Release Notes	Contains installation details and other information that was not available until immediately before the final release of Sun WorkShop 6. This document complements the information that is available in the component readme files.
Forte Developer 6 / Sun WorkShop 6	*Analyzing Program Performance With Sun WorkShop 6*	Explains how to use the new Sampling Collector and Sampling Analyzer (with examples and a discussion of advanced profiling topics) and includes information about the command-line analysis tool `er_print`, the LoopTool and LoopReport utilities, and UNIX profiling tools `prof`, `gprof`, and `tcov`.
	Debugging a Program With dbx	Provides information on using dbx commands to debug a program with references to how the same debugging operations can be performed using the Sun WorkShop Debugging window.
	Introduction to Sun WorkShop	Acquaints you with the basic program development features of the Sun WorkShop integrated programming environment.

TABLE P-3 Related Sun WorkShop 6 Documentation by Document Collection *(Continued)*

Document Collection	**Document Title**	**Description**
Forte™ C 6 / Sun WorkShop 6 Compilers C	*C User's Guide*	Describes the C compiler options, Sun-specific capabilities such as pragmas, the `lint` tool, parallelization, migration to a 64-bit operating system, and ANSI/ISO-compliant C.
Forte™ C++ 6 / Sun WorkShop 6 Compilers C++	C++ *Library Reference*	Describes the C++ libraries, including C++ Standard Library, Tools.h++ class library, Sun WorkShop Memory Monitor, Iostream, and Complex.
	C++ *Migration Guide*	Provides guidance on migrating code to this version of the Sun WorkShop C++ compiler.
	C++ *Programming Guide*	Explains how to use the new features to write more efficient programs and covers templates, exception handling, runtime type identification, cast operations, performance, and multithreaded programs.
	C++ *User's Guide*	Provides information on command-line options and how to use the compiler.
	Sun WorkShop Memory Monitor User's Manual	Describes how the Sun WorkShop Memory Monitor solves the problems of memory management in C and C++. This manual is only available through your installed product (see `/opt/SUNWspro/docs/index.html`) and not at the `docs.sun.com` Web site.
Forte™ for High Performance Computing 6 / Sun WorkShop 6 Compilers Fortran 77/95	*Fortran Library Reference*	Provides details about the library routines supplied with the Fortran compiler.

TABLE P-3 Related Sun WorkShop 6 Documentation by Document Collection *(Continued)*

Document Collection	Document Title	Description
	Fortran Programming Guide	Discusses issues relating to input/output, libraries, program analysis, debugging, and performance.
	Fortran User's Guide	Provides information on command-line options and how to use the compilers.
	FORTRAN 77 Language Reference	Provides a complete language reference.
	Interval Arithmetic Programming Reference	Describes the intrinsic `INTERVAL` data type supported by the Fortran 95 compiler.
Forte™ TeamWare 6 / Sun WorkShop TeamWare 6	*Sun WorkShop TeamWare 6 User's Guide*	Describes how to use the Sun WorkShop TeamWare code management tools.
Forte Developer 6 / Sun WorkShop Visual 6	*Sun WorkShop Visual User's Guide*	Describes how to use Visual to create C++ and Java™ graphical user interfaces.
Forte™ / Sun Performance Library 6	*Sun Performance Library Reference*	Discusses the optimized library of subroutines and functions used to perform computational linear algebra and fast Fourier transforms.
	Sun Performance Library User's Guide	Describes how to use the Sun-specific features of the Sun Performance Library, which is a collection of subroutines and functions used to solve linear algebra problems.
Numerical Computation Guide	*Numerical Computation Guide*	Describes issues regarding the numerical accuracy of floating-point computations.
Standard Library 2	*Standard C++ Class Library Reference*	Provides details on the Standard C++ Library.
	Standard C++ Library User's Guide	Describes how to use the Standard C++ Library.
Tools.h++ 7	*Tools.h++ Class Library Reference*	Provides details on the `Tools.h++` class library.
	Tools.h++ User's Guide	Discusses use of the C++ classes for enhancing the efficiency of your programs.

TABLE P-4 describes related Solaris documentation available through the `docs.sun.com` Web site.

TABLE P-4 Related Solaris Documentation

Document Collection	Document Title	Description
Solaris Software Developer	*Linker and Libraries Guide*	Describes the operations of the Solaris link-editor and runtime linker and the objects on which they operate.
	Programming Utilities Guide	Provides information for developers about the special built-in programming tools that are available in the Solaris operating environment.

What’s New in Sun WorkShop 6

This document describes the new features of the Sun WorkShop™ 6 compilers and tools. Tables summarize new product features, and some product features are described in additional detail in the sections that follow each table.

Why Upgrade to Sun WorkShop 6?

The following key features are the highlights of this release:

- Additional ANSI/ISO C++ compliance
- Fortran 95 compiler with OpenMP parallelization directives
- Support for the UltraSPARC™ III instruction set architecture
- Easier-to-use programming environment
- New Performance Analysis tool
- Fortran 95 interval arithmetic
- Installation improvements
- Manuals, man pages, READMEs, and online help in HTML

Sun WorkShop 6 C Compiler

TABLE 1 lists the new features available with the release of the Sun WorkShop 6 C compiler. These features enhance the capabilities of the C compiler and `lint` source-code checker.

TABLE 1 Sun WorkShop 6 C Compiler

Feature	Description
`__func__`	The C compiler predefines a static, constant, char array named `__func__` for every function definition. The array is initialized with the name of the function and can be used anywhere a static function scope array can be used, such as when printing the name of the enclosing function.
Variable argument macro	The C preprocessor accepts a variable number of arguments for a `#define` macro. If the macro definition includes an ellipsis as part of the identifier list, then there will be more arguments when the macro is invoked than there were parameters in the macro definition.
`SUNW_MP_THR_IDLE`	You can use the `SUNW_MP_THR_IDLE` environment variable to control whether a thread continues to use system resources after it finishes its task or it "sleeps."
large arrays	The C compiler supports significantly larger array objects. For specifics, see Appendix A of the *C User's Guide*.
`-errchk=locfmtchk`	`lint` accepts a new flag, `-errchk=locfmtchk`, which checks for `printf`-like format strings during the first pass of `lint`.
new `lint` directive (`PRINTFLIKE(`*n*`))`	`lint` accepts a new directive that identifies all calls to the `printf()` family through a pointer. All such calls through the pointer can be checked for argument consistency by `lint`.
`-errwarn=t`	The C compiler and the `lint` source code checker support a new option, `-errwarn=t`, which causes the compiler to exit with a failure status if any of the specified warnings are issued.
`-errchk`	The `-errchk` option of `lint` has a new value `signext,` which you can use in conjunction with the `longptr64` option: `-errchk=longptr64,signext`. This new option warns about sign extension in order to facilitate migration to the 64-bit development environment.

TABLE 1 Sun WorkShop 6 C Compiler *(Continued)*

Feature	Description
`-xchar_byte_order`	The `-xchar_byte_order` option produces an integer constant by placing the characters of a multi-character character-constant in the specified byte order.
`-xinline`	The `-xinline` option accepts two new values: `%auto` and `no%`*function_name*. The `%auto` value takes effect at the `-x04` level of optimization and allows the compiler to automatically inline functions in addition to the listed function for `-xinline`. The `%no` *function_name* value tells the compiler to not inline the function *function_name*.
`-xmemalign`	The C compiler offers a new option named `-xmemalign`. This option controls code generated for potentially misaligned memory accesses and controls program behavior in the event of a misaligned access.
`-xprefetch`	You can use the `sun_prefetch.h` header file and the `-xprefetch` option to specify explicit prefetch instructions.
`-xvector`	

Sun WorkShop 6 C++ Compiler

TABLE 2 lists the new features available with the release of the Sun WorkShop™ 6 C++ compiler. Some of these features are described more completely in the sections following the table.

TABLE 2 Sun WorkShop 6 C++ Compiler

Feature	Description
Partial Specialization	A template can be partially specialized, meaning that only some of the template parameters are specified or that one or more parameters are limited to certain categories of type.
Explicit Function Template Argument	If a template argument cannot be deduced from the function arguments, you can specify it explicitly using the syntax `f<`*template args*`>(`*function args*`)`.

Non-Type Function Template Parameters	This release supports non-type function template parameters, such as:
	`template<int I> void foo( int a[I] ) { ... }`
	`template<int I> void foo( mytype<I> m ) { ... }`

TABLE 2 Sun WorkShop 6 C++ Compiler *(Continued)*

Feature	Description
Member Templates	In standard mode, classes and class templates can have templates as members.
Definitions-Separate Template Organization Restriction Removed	The compiler no longer has a restriction against "definitions-separate template organization" for `-instances != extern` (that is, `-instances=explicit`, `-instances=global`, `-instances=semiexplicit`, or `-instances=static`). Regardless of the `-instances` setting, by default the compiler now includes separate source files in the search for definitions.
Prefetch Instructions	You can use `-xprefetch` in conjunction with the header file `<sun_prefetch.h>` to specify prefetch instructions on those architectures that support prefetch, such as the UltraSPARC II instruction set architecture (`-xarch=v8plus`, `v8plusa`, `v8plusb`, `v9`, `v9a`, or `v9b`).
Extern Inline Functions	This version of the compiler allows extern inline functions. If there is any local static data in an inline function, only one copy of the static data is used in all compilation units. However, the addresses of inline functions taken in different translation units will not compare as equal.
Ordering of Static Variable Destruction	The standard has defined the order of destruction of objects with static storage duration more fully; static objects must be destroyed in the reverse order of their construction. Previous language definitions left some aspects unspecified.
Sub-Aggregate Initialization	When using brace-initialization of class objects (for types where brace-initialization is allowed), the C++ standard permits a member which is itself an aggregate class to be initialized by a value of its own type.
Using Your Own C++ Standard Library	By specifying the `-library=no%Cstd` option, you can use your own version of the C++ standard library, instead of the version supplied with the compiler.
Cache Versioning	The C++ compiler has the ability to detect cache version differences and issue the appropriate error message.
Restrictions on Bitfield Size Removed	The restriction on the size of a bitfield to 32 or less is removed. Bitfields can be any size.

TABLE 2 Sun WorkShop 6 C++ Compiler *(Continued)*

Feature	Description
Warnings About Conversions Between Pointer-To-Function and `void*`	Previously, the compiler issued warnings about conversions between pointer-to-function and `void*`. The compiler only issues these warnings when you use the `+w2` option.

New and Changed Options	The following list shows the new and changed options.
	• New Options:
	`-xcrossfile`
	`-Bsymbolic`
	`-features=[no%]strictdestrorder`
	`-template=extdef`.
	• Changed Options:
	`-fast`
	`-library=[no%]Cstd`, `+p`
	`-ptr`, `-xprefetch`
	
	For more information, see the entry for each option in the *C++ User's Guide*. To access this guide in HTML, point your browser to `file:/opt/SUNWspro/docs/index.html`.

Partial Specialization

A template can be fully specialized, meaning that an implementation is defined for specific template arguments. See the following code example.

```
template<class T, class U> class A { ... }; //primary template
template<> class A<int, double> { ... }; //specialization
```

A template can also be partially specialized, meaning that only some of the template parameters are specified, or that one or more parameters are limited to certain categories of type. The resulting partial specialization is itself still a template. The following examples use the previous primary template.

- Special template definition for cases when the first template parameter is type `int`. See the following code example.

  ```
  template<classU> class A<int> { ... };
  ```

- Special template definition for cases when the first template parameter is any pointer type. See the following code example.

  ```
  template<class T, class U> class A<T*> { ... };
  ```

- Special template definition for cases when the first template parameter is pointer-to-pointer of any type, and the second template parameter is type `char`. See the following code example.

  ```
  template<class T> class A<T**, char> { ... };
  ```

Explicit Function Template Argument

If a template argument cannot be determined from the function arguments, you can now explicitly specify the template argument using the syntax `f<`*template args*`>(`*function args*`)`. See the following code example.

```
template<class Mytype> Mytype* construct(float, float);
...
int* x = construct<int>(a, b);
```

Non-Type Function Template Parameters

This release supports non-type function template parameters, as illustrated in the following code example.

```
template<int I> void foo( int a[I] ) { ... }
template<int I> void foo( mytype<I> m ) { ... }
```

This release does not allow expressions involving non-type template parameters in the function parameter list, as illustrated in the following code example.

```
// these are not supported
template<int I> void foo( mytype<2*I> ) { ... }
template<int I, int J> void foo( int a[I+J] ) { ... }
```

Member Templates

In standard mode, classes and class templates can have templates as members, as illustrated in the following code example.

```
template <class T1>
class OuterClass {
public:
    // class member template
    template <class T2>
        class MemberClass
    {
        T2 MCmember;
        T1 OCmember;
    };
    template<class T3> operator T3() { ... }
...
};
```

Note – Member templates are not supported in compatibility mode (`-compat[=4]`).

Definitions-Separate Template Organization Restriction Removed

The compiler no longer has a restriction against "definitions-separate template organization" for `-instances != extern` (that is, `-instances=explicit`, `-instances=global`, `-instances=semiexplicit` or `-instances=static`). Regardless of the `-instances` setting, the compiler will now, by default, include separate source files in the search for definitions.

To turn this restriction back on, use the `-template=no%extdef` option. Note, however, that when the `-template=no%extdef` option is specified, the compiler does not search for separate source files even with `-instances=extern`.

Ordering of Static Variable Destruction

The standard has defined the order of destruction of objects with static storage duration more fully; static objects must be destroyed in the reverse order of their construction. Previous language definitions left some aspects unspecified.

This stricter ordering is implemented for standard mode only. In compatibility mode (`-compat[=4]`), the order of destruction is implemented as before.

If your program depends on a particular order of destruction and worked with an older compiler, the order required by the standard might break the program in standard mode. The `-features=no%strictdestrorder` command option disables the strict ordering of destruction.

Sub-Aggregate Initialization

When using brace-initialization of class objects (for types where brace-initialization is allowed), the C++ standard permits a member that is itself an aggregate class to be initialized by a value of its own type. See the following code example.

```
struct S {  // an aggregate type
int i, j;
};
struct T { // an aggregate type
S s; // aggregate member
int k;
};
T t1 = { {1, 2}, 3 }; // traditional initialization
S s1 = { 1, 2 };
T t2 = { s1, 3 }; // sub-aggregate initialization
```

Using Your Own C++ Standard Library

If you want to use your own version of the C++ standard library instead of the version supplied with the compiler, you can do so by specifying the `-library=no%Cstd` option. This option prevents finding any of the following headers:

```
<algorithm> <bitset> <complex> <deque> <fstream> <functional>
<iomanip> <ios> <iosfwd> <iostream> <istream> <iterator> <limits>
<list> <locale> <map> <memory> <numeric> <ostream> <queue> <set>
<sstream> <stack> <stdexcept> <streambuf> <string> <strstream>
<utility> <valarray> <vector>
```

When `-library=no%Cstd` is specified, the `libCstd` library, which implements those headers, is not automatically linked with your program. To use any of the features declared in the above headers, you must use the `-I` option to point to the directory where the replacement headers are located, and you must link your program with a library or set of object files containing the implementation of the replacement headers.

You cannot reliably replace only a portion of the headers listed above, nor can you reliably link `libCstd` with all or part of another library implementation. For example, you cannot replace only the string classes and use `libCstd` for everything else. Either use the library supplied with the compiler, or replace all of the functionality listed above.

The remaining headers (`<exception>`, `<new>`, `<typeinfo>`, and all the headers inherited from C) are integral to the compiler itself or to Solaris, and are not affected by the `-library=no%Cstd` option. Linking of the library `libCrun` also is not affected by the `-library=no%Cstd` option.

There is no mechanism to replace any of the functionality of `libCrun`. If you replace the standard library, the code must be compiled with the versions of `<exception>`, `<new>`, and `<typeinfo>` supplied with the compiler. In standard mode (the default mode) C++ programs must be linked with `libCrun`.

Note – This option is available to "use at your own risk." Using your own version of the C++ standard library might not produce optimal results.

Cache Versioning

The C++ compiler now has the ability to detect cache version differences and issue the appropriate error message. The compiler marks each template cache directory with a version string that uniquely identifies the template cache version. Subsequent releases of the compiler will also use cache version strings, although these versions may be different from the current version.

This compiler and future compilers will detect the version strings from within the cache directories and issue an error as appropriate. For example, a future compiler that uses a different template cache version string and processes a cache directory produced by this release of the compiler might issue the following error.

```
SunWS_cache: Error: Database version mismatch
/SunWS_cache/CC_version
```

Similarly, this release of the compiler issues an error if it encounters a cache directory produced by a future compiler.

The template cache directories produced by the Sun WorkShop C++ compiler 5.0 compiler are not versioned. However, the Sun WorkShop 6 C++ compiler processes these cache directories without an error or a warning. These cache directories are converted to the cache directory format used by the Sun WorkShop 6 C++ compiler.

A template cache directory produced by the Sun WorkShop 6 C++ compiler or later releases cannot be used by the Sun WorkShop C++ compiler 5.0. The Sun WorkShop C++ compiler 5.0 is not capable of recognizing format differences and it will issue an assertion.

Sun WorkShop 6 Fortran Compilers

Sun WorkShop 6 includes the Sun WorkShop™ 6 Fortran 77 and Sun WorkShop™ 6 Fortran 95 compilers.

Sun WorkShop 6 Fortran 77 Compiler

TABLE 3 lists the new features available with the release of the Sun WorkShop 6 Fortran 77 compiler. Some of these features are described more completely in "New Fortran Compiler Features" on page 22.

TABLE 3 Sun WorkShop 6 Fortran 77 Compiler

Feature	Description
`FORM="BINARY"` I/O Extension	Specifying this new option in an `OPEN(...)` statement causes the file to be treated as a sequential binary (unformatted) file with no record marks. This enables data to be written and read as a continuous stream of bytes, and provides compatibility with other vendor systems. It is implemented in both the Fortran 95 and Fortran 77 compilers.
Debugging Optimized Code	The restrictions on compiling with `-g` have been relaxed so that it is now possible to compile at `-O4` and `-O5` or any of the parallelization flags (`-parallel`, `-explicitpar`, `-autopar`) with debugging (`-g`).

TABLE 3 Sun WorkShop 6 Fortran 77 Compiler *(Continued)*

Feature	Description	
New Command-Line Flags	The following new command-line flags appear in this release of `f77` (see the `f77`(1) man page):	
	`-aligncommon`	Aligns common block elements to specified byte boundaries
	`-r8const`	Promotes single-precision data constants to `REAL*8`
	`-xmemalign`	Specifies general alignment of data elements
Expanded Command-Line Flags	The following `f77` command-line flags have been expanded (see the `f77`(1) man page):	
	`-fast`	Sets `-O5, -fsimple=2, -xvector=yes, -pad=common`
	`-xprefetch`	Enables explicit pragma prefetch directives to force generation of prefetch instructions on UltraSPARC platforms
	`-xtypemap`	Includes an expanded set of possible data type specifications
Cray-Style Directives	`AUTOSCOPE` qualifier added to Cray-style parallelization directives.	
Licensing	The parallelization features of the Fortran 77 compiler require a Sun WorkShop HPC license.	
Hyper-Linked Diagnostics	`f77` error messages in the Sun WorkShop Building window now have hyperlinks to help pages that explain the messages.	

Sun WorkShop 6 Fortran 95 Compiler

TABLE 4 lists the new features available with the release of the Sun WorkShop 6 Fortran 95 compiler. Some of these features are described more completely in "New Fortran Compiler Features" on page 22.

TABLE 4 Sun WorkShop 6 Fortran 95 Compiler

Feature	Description
Compliance	The Fortran 95 compiler is fully compliant with the Fortran 95 standard.
New Command	The Fortran 95 compiler is invoked by both the `f90` and `f95` command. The `f95` command is new. `f90` is equivalent to `f95`.

TABLE 4 Sun WorkShop 6 Fortran 95 Compiler *(Continued)*

Feature	Description
File Extensions	The compiler will accept free-format source files with `.f95` and `.F95` extensions as well as `.f90` and `.F90.`
`FORM="BINARY"` I/O Extension	Specifying this new option in an `OPEN(...)` statement causes the file to be treated as a sequential binary (unformatted) file with no record marks. The enables data to be written and read as a continuous stream of bytes, and provides compatibility with other vendor systems. It is implemented in both the Fortran 95 and Fortran 77 compilers. See the *FORTRAN 77 Language Reference.*
Debugging Optimized Code	The restrictions on compiling with `-g` have been relaxed so that it is now possible to compile at `-O4` and `-O5` or any of the parallelization flags (`-parallel`, `-explicitpar`, `-autopar`) with debugging (`-g`).
`f77` Flags	Most of the `f77` compiler flags are now implemented in `f95`. See the `f95`(1) man page for details. These include:
	`-erroff` Turns off selected error messages
	`-errtags` Displays error messages with tags
	`-ext_names` Creates external names with or without underscores
	`-fpp` Specifies source code preprocessor
	`-loopinfo` Shows which loops are parallelized
	`-sbfast` Produces browser table information
	`-silent` Suppresses compiler messages
	`-U` Allows lowercase and uppercase
	`-u` Implies `IMPLICIT NONE`
	`-xcrossfile` Enables optimization across files
	`-xF` Allows function-level reordering for Analyzer
	`-xinline` Compiles functions inline
	`-xtypemap` Specified default data sizes

TABLE 4 Sun WorkShop 6 Fortran 95 Compiler *(Continued)*

Feature	Description	
New Flags	The following new flags are implemented in `f95`:	
	`-aligncommon`	Aligns common block elements to specified byte boundaries
	`-mp=openmp`	Accepts OpenMP directives
	`-r8const`	Promotes single-precision constants to `REAL*8`
	`-xia`	Enables processing of `INTERVAL` data types (recommended)
	`-xinterval`	Enables processing of `INTERVAL` data types
	`-xmemalign`	Specifies general alignment of data elements
	`-xrecursive`	Allows recursive calls without `RECURSIVE` attribute
Expanded Flags	The following `f95` command-line flags have been expanded (see the `f95`(1) man page):	
	`-fast`	Sets `-O5` `-fsimple=2` `-xvector=yes` `-pad=common`
	`-xprefetch`	Enables explicit pragma prefetch directives to force generation of prefetch instructions on UltraSPARC platforms
	`-xtypemap`	Includes an expanded set of possible data type specifications
OpenMP	This release of Fortran 95 implements the OpenMP interface for explicit parallelization, including a set of source code directives, runtime library routines, and environment variables. See the *Fortran User's Guide.*	
Cray-Style Directives	`AUTOSCOPE` qualifier added to Cray-style parallelization directives.	
Interval Arithmetic	This release of Fortran 95 implements extensions to support intrinsic `INTERVAL` data types.	
Licensing	The parallelization features of the Fortran 95 compiler require a Sun WorkShop HPC license.	
Hyper-Linked Diagnostics	Sun WorkShop Building window now interprets `f95` error messages as live links into the online help.	

New Fortran Compiler Features

The following sections describe some of the new Fortran compiler features in greater detail.

Effect of `FORM="BINARY"` on I/O Operations

- `WRITE` statement: Data is written to the file in binary, with as many bytes transferred as there is specified in the output list.
- `READ` statement: Data is read into the variables on the input list, with as many bytes transferred as demanded by the list. Because there are no record marks on the file, there will be no "end-of-record" error detection. The only error detected is end-of-file, or abnormal system errors.
- `INQUIRE` statement: `INQUIRE` on a file opened with `FORM="BINARY"` returns the following information:

  ```
  FORM="BINARY"
  ACCESS="SEQUENTIAL"
  SEQUENTIAL="YES"
  DIRECT="NO"
  FORMATTED="NO"
  UNFORMATTED="YES"
  RECL= and NEXTREC= are undefined.
  ```
- `BACKSPACE` statement: Not allowed — returns an error.
- `ENDFILE` statement: Truncates file at current position, as usual.
- `REWIND` statement: Repositions file to beginning of data, as usual.

OpenMP

Fortran 95 only —This release of Fortran 95 implements the OpenMP interface for explicit parallelization, including a set of source code directives, runtime library routines, and environment variables. Preliminary documentation is available in the OpenMP README. The OpenMP specifications can be viewed at `http://www.openmp.org/`

A summary of all directives accepted by the Fortran compilers, including OpenMP, can be found in Appendix E of the *Fortran User's Guide*. See also the *Fortran Programming Guide* for additional information on the parallelization features of the Fortran compilers.

Note – The parallelization features of the Fortran compilers require a Sun WorkShop HPC license.

Interval Arithmetic Extensions

Fortran 95 only —This release of Fortran 95 includes interval arithmetic extensions. See the section "Sun WorkShop 6 Fortran 95 Interval Arithmetic."

Hyper-Linked Compiler Diagnostics

When you use Sun WorkShop to build and compile applications, `f77` and `f95` diagnostic messages in the Building window now have hyperlinks to help pages. Clicking on the error message launches a help browser with additional information about the specific error diagnostic.

Fortran Platforms

The Fortran compilers in this Sun WorkShop 6 release support only versions 2.6, 7, and 8 of the Solaris *SPARC™ Platform Edition* Operating Environment. For the Solaris Intel IA-32 platform, Sun has discontinued development of Fortran compilers and the Sun Performance Library. Sun does not offer Forte™ for High Performance Computing or the Forte™ Fortran Desktop Edition (formerly known as Sun Performance WorkShop Fortran) for the Solaris Intel IA-32 platform in this release. You can contact The Portland Group (`http://www.pgroup.com`) for information about their line of software development tools for the Solaris IA-32 platform.

Sun WorkShop 6 Fortran 95 Interval Arithmetic

Support for intrinsic `INTERVAL` data types is a new feature in the Sun WorkShop 6 Fortran 95 compiler.

Two new compiler flags, `-xia` and `-xinterval`, tell the compiler to recognize interval-specific languages extensions and generate the code to implement interval instructions.

What Is Interval Arithmetic?

Interval arithmetic is used to evaluate arithmetic expressions over sets of numbers contained in intervals. An interval is the set of all real numbers between and including the interval's lower and upper bound. Any interval arithmetic result is a new interval that is guaranteed to contain the set of all possible resulting values.

With Sun WorkShop 6 Fortran 95, it is a simple matter to write interval programs to compute rigorous bounds on the value of arithmetic expressions:

- Declare variables to be type `INTERVAL`.
- Write normal Fortran code using the intrinsic `INTERVAL` functions and operators, relational operators, and format edit descriptors.
- Compile the code using the `-xia` command-line option.

To achieve the best results, use existing interval algorithms that compute narrow width interval results. Devising algorithms to compute narrow interval results is the topic of interval analysis.

Why Is Interval Arithmetic Important?

Interval arithmetic is important for the following reasons:

- Interval arithmetic can be used to perform machine computations with guaranteed bounds on errors from all sources, including input data errors, machine rounding, and their interactions.
- Interval algorithms can be developed that solve nonlinear problems, such as the solution to nonlinear systems of equations and nonlinear programming.

As intervals become more widely used, libraries of interval solvers will be used routinely to compute sharp interval solutions to linear and nonlinear problems, while taking into account all sources of error. With these libraries, scientists, engineers, and developers of commercial applications will be able to write programs to solve problems that are currently out of reach.

Where Can I Get More Information?

See the *Interval Arithmetic Programming Reference* or the list of online resources in the Interval Arithmetic README.

Sun WorkShop 6 dbx

TABLE 5 lists the new features available with the release of Sun WorkShop 6 dbx.

TABLE 5 Sun WorkShop 6 dbx

Feature	Description
$firedhandlers ksh variable	The read only ksh variable $firedhandlers has been added. This variable can be used in conjunction with the delete and handler commands as an alternative to the clear command. For more information, see Variables, delete Command, and handler Command in the Using dbx Commands section of the Sun WorkShop online help.
Partial clearing of breakpoints	The clear command now facilitates partial clearing of In Class, In Method, and In Function breakpoints. For more information, see clear Command in the Using dbx Commands section of the Sun WorkShop online help.
Trace output	Output of traces can be redirected to files. For more information, see trace Command in the Using dbx Commands section of the Sun WorkShop online help.
New dbx environment variables	• The new dbx environment variable stack_find_source controls whether dbx automatically moves up the call stack to a frame with debuggable source code when the program stops. For more information, see stack_find_source Environment Variable in the Using dbx Commands section of the Sun WorkShop online help. • The new dbx environment variable proc_exclusive_attach controls whether dbx can attach to a process that is under the control of another debugger or debugging tool. For more information, see proc_exclusive_attach Environment Variable in the Using dbx Commands section of the Sun WorkShop online help. • The new dbx environment variable step_granularity controls whether the step and next commands work on statements or lines or source code. For more information, see step_granularity Environment Variable in the Using dbx Commands section of the Sun WorkShop online help.

TABLE 5 Sun WorkShop 6 `dbx` *(Continued)*

Feature	Description
	• The new `dbx` environment variable `mt_scalable` helps debugging multithreaded applications with many LWPs (lightweight processes) by reducing resource usage. For more information, see mt_scalable Environment Variable in the Using dbx Commands section of the Sun WorkShop online help. • The new `dbx` environment variable `rtc_error_stack` determines whether stack traces show frames corresponding to RTC internal mechanisms. For more information, see rtc_error_stack Environment Variable in the Using dbx Commands section of the Sun WorkShop online help. • The new `dbx` environment variable `rtc_inherit` determines whether runtime checking is enabled on child processes that are executed from the debugged program. For more information, see rtc_inherit Environment Variable in the Using dbx Commands section of the Sun WorkShop online help. • The new `dbx` environment variable `rtc_use_traps` enables a workaround for the eight megabyte code limitation on runtime checking. For more information, see rtc_use_traps Environment Variable in the Using dbx Commands section of the Sun WorkShop online help.
Breakpoints in C++ member functions	Breakpoints in C++ inlined member functions work correctly. This includes proper evaluation of handler conditions (`-if`).
LWP-related commands now always available	LWP-related commands are now always available, not just when you are debugging a multithreaded application. For more information, see lwp Command and lwps Command in the Using dbx Commands section of the Sun WorkShop online help.
Interrupting a running process	When Control-C does not seem to stop a hung process, the third consecutive Control-C will force the process to stop by other means. See Interrupting a Running Process in the Using dbx Commands section of the Sun WorkShop online help.
Full line number information	Full line number information is now recorded when you compile with the `-g` and `-O` options. You can now use the `step` command and `next` command to step through optimized code, although the current line will jump forward and back due to code scheduling. The values of variables printed from within dbx continue to be unreliable in optimized code.

Sun WorkShop 6

TABLE 6 lists the new features available with the release of Sun WorkShop 6. Some of the features are described in greater detail in the sections following the table.

TABLE 6 Sun WorkShop 6 New Features

Feature	Description
Text Editing	
Balloon expression evaluator	The balloon expression evaluator instantly shows you the current value of the expression at which your cursor is pointing in your text editor.
NEdit and Vim text editors	The NEdit and Vim text editors are new integrated text editors for this release.
Debugging a Program	
Button Editor	The Button Editor now allows you to customize the toolbars in your editor window and the Debugging window, in addition to adding, removing, and editing buttons in the Custom Button window.
Debugging window	The Data History pane has been replaced with tabs for Data History and Program I/O, and an optional Data Display tab. The radio buttons for switching the Sessions and Threads panes have been replaced by tabs for Sessions, Threads, and Breakpoints.
Debugging Options dialog box	The Category list box has been replaced with tabs at the right side of the dialog box for switching among the categories of options.
New debugging options	You now choose to direct program input or output to the Program I/O tab in the Debugging window. You can now choose to have the call stack pop to the first debuggable source code if execution stops in a function in your program that is not debuggable source code. You can now choose to set the step granularity to Line, so that a single `next` command steps through a line regardless of the number of statements it includes. By default, the State, Stopped In, Evaluation Context, and number of sessions information is no longer displayed below the toolbar, but you can choose to display this information.

TABLE 6 Sun WorkShop 6 New Features *(Continued)*

Feature	Description
	You can choose to not have the debugger warn you if the `main()` module of your program is not compiled with debugging information.
	You can choose to have the Data Display window shown as a tab in the Debugging window or as a separate window.
	You can delay the loading of modules compiled with `-xs` until the debugging information for these modules is needed, rather than having the modules loaded during the startup of the debugging session.
	If you want to debug processes with a large number (up to 300) LWPs (lightweight processes), you can now set debugging to be conservative in its resource usage when doing so.
	You can choose not to have the debugger check that `dbx` has exclusive control of the program being debugged.
	You can choose to allow the debugger to exclude certain shared libraries that are critical to `dbx` functionality.
Removed debugging option	The option to allow watchpoints in multithreaded programs has been removed from the Debugging Options dialog box.
Working With Projects	
Projects and worksets	This release of Sun WorkShop uses projects to track the files, programs, and targets associated with your development projects and to build your programs without your needing to write a makefile.
New Man Pages	
`makeprd`(1)	Sun WorkShop project file builder
`nedit`(1)	Motif user interface style text editor
`vim`(1)	Vi improved; a text editor for programmers
`xemacs`(1)	Emacs: The Next Generation

Text Editing

The following two sections describe the new Sun WorkShop 6 text editing features.

Balloon Expression Evaluator

The balloon expression evaluator instantly shows you the current value of the expression at which your cursor is pointing in your text editor. You can also see the type of expression and dereference the pointer through the balloon expression evaluator feature. For more information, see Using the Balloon Expression Evaluator in the Text Editing section of the online help.

NEdit and Vim Text Editors

Text editors are the center of the Sun WorkShop integrated development tool set that includes building, debugging, and browsing. The Sun WorkShop programming environment makes it possible to evaluate expressions, set breakpoints, and step through functions from your text editor.

NEdit, a plain-text editor with a graphical user interface for X/Motif systems, and Vim, an improved version of the vi standard text editor on UNIX systems, are new integrated text editors in this release. The following is a list of integrated text editors provided with Sun WorkShop 6:

- NEdit (version 5.1)
- XEmacs (version 20.4 or compatible versions)
- GNU Emacs (version 19.28 or compatible versions)
- Vi
- Vim (version 5.3 with graphical user interface option)

For more information about each editor's options, see:

- The online documentation available from the Help menu in the editor's menu bar
- Text Editor Options Dialog Box in the Text Editing section of the online help

Debugging a Program

The following sections describe new program debugging features.

Button Editor

The Button Editor now allows you to customize the toolbars in your editor window and the Debugging window, in addition to adding, removing, and editing buttons in the Custom Button window.

For more information, see Toolbar Options.

Debugging Window

The following sections describe changes made in the Debugging window.

Session Status and Context Information

By default, the State, Stopped In, Evaluation Context, and Number of Sessions information no longer appears below the toolbar. If you want to display this information, you can select Show 3 Line Context/Status Area at Top of Display on the Window Layout tab in the Debugging Options dialog box. For more information, see Displaying Session Status and Context Information in the Using the Debugging Window section of the Sun WorkShop online help.

Data History Pane

The Data History pane has been replaced with tabs for Data History and Program I/O, and an optional Data Display tab:

- The Data History tab displays the Data History pane. For more information, see Data History Tab in the Using the Debugging WIndow section of the Sun WorkShop online help.
- The Program I/O tab displays program input and output within the Debugging window rather than in a separate Program Input/Output window. (For more information, see Program I/O Tab in the Using the Debugging Window section of the Sun WorkShop online help.) This is now the default behavior, but you can use the Program output section of the Debugging Output tab of the Debugging Options dialog box to choose where you want to direct program input and output. For more information, see Redirecting a Program's Input/Output in the Using the Debugging Window section of the Sun WorkShop online help.

- The Data Display tab is included (instead of a separate Data Display window) if you select Tab in Debug Window in the Data Display Window Shown As section on the Data Display Window tab in the Debugging Options dialog box. For more information, see Data Display Tab and Choosing How to Show the Data Display in the Using the Debugging Window section of the Sun WorkShop online help.

Sessions and Threads Panes

The radio buttons for switching the Sessions and Threads panes have been replaced by tabs for Sessions, Threads, and Breakpoints:

- The Sessions tab displays the Sessions pane. When you press the right mouse button over this tab, a popup menu is displayed. For more information, see Sessions Tab in the Using the Debugging Window section of the Sun WorkShop online help.
- The Threads tab displays the Threads pane. For more information, see Threads Tab in the Using the Debugging Window section of the Sun WorkShop online help.
- The Breakpoints tab displays a scrolling list of breakpoints and tracepoints assigned in your program. When you press the right mouse button over this tab, a popup menu is displayed that lets you enable, disable, delete and show source instantly for each breakpoint. The popup menu includes an Add item that displays the separate Breakpoints window in which you add, enable, disable, change, and delete breakpoints. For more information, see Breakpoints Tab in the Using the Debugging Window section of the Sun WorkShop online help.

Debugging Options Dialog Box

The Debugging Options dialog box contains the following new features.

Category Tabs

The Category list box in the Debugging Options dialog box has been replaced with tabs at the right side of the window for switching among the categories of options.

New Program Output Option

You now direct program input or output to the Program I/O tab in the Debugging window. As before, you can also direct it to a separate Program I/O window, to the `dbx` Commands window, or to a custom `pty`. For more information, see Redirecting a Program's Input/Output in the Using the Debugging Window section of the Sun WorkShop online help.

New Call Stack Option

During debugging, if your program was not compiled with `-g`, execution might stop in a function in the program that is not debuggable source code. You can now choose to have the call stack pop to the first debuggable source code in this case. For more information, see Going Up the Stack When Execution Stops in the Using the Debugging Window section of the Sun WorkShop online help.

New Stepping Option

By default, the step granularity for debugging is set to Statement, so that if more than one statement is included in a source code line, it requires the same number of commands to step through that line. You can now choose to set the step granularity to Line, so that a single next command steps through a line regardless of the number of statements it includes. For more information, see Setting Step Granularity in the Using the Debugging Window section of the Sun WorkShop online help.

New Window Layout Option

By default, the State, Stopped In, Evaluation Context, and number of sessions information is no longer displayed below the toolbar. If you want to display this information, you can select Show 3 Line Context/Status Area at Top of Display on the Window Layout tab in the Debugging Options dialog box. For more information, see Displaying Session Status and Context Information in the Using the Debugging Window section of the Sun WorkShop online help.

New Window Behavior Option

By default, the debugger warns you if the `main ()` module of your program is not compiled with debugging information. You can choose to not have this warning displayed. For more information, see Being Warned If Your main () Module is Not Compiled With Debugging Information in the Using the Debugging Window section of the Sun WorkShop online help.

New Data Display Option

You can choose to have the Data Display window shown as a tab in the Debugging window or as a separate window. For more information, see Choosing How to Show the Data Display in the Using the Debugging Window section of the Sun WorkShop online help.

New Debugging Performance Option

You can delay the loading of modules compiled with `-xs` until the debugging information for these modules is needed, rather than having the modules loaded during the startup of the debugging session. This may shorten the debugging startup time when modules have been compiled with `-xs`. By default, this debugging option is set to on. For more information, see Delaying Loading of Modules Compiled with -xs in the Using the Debugging Window section of the Sun WorkShop online help.

New Forks and Threads Option

If you want to debug processes with a large number (up to 300) of LWPs (lightweight processes), you can now set debugging to be conservative in its resource usage. When doing so, debugging performance might be slowed. For more information, see Debugging a Large Number of LWPs in the Using the Debugging Window section of the Sun WorkShop online help.

New Advanced Options

By default, the debugger checks that `dbx` has exclusive control of the program being debugged. It prevents `dbx` from attaching to the process if another tool is already attached to the process. You can now turn off this behavior. For more information, see Checking that dbx has Exclusive Control of the Program in the Using the Debugging Window section of the Sun WorkShop online help.

By default, the debugger disallows the exclusion of certain shared libraries that are critical to `dbx` functionality. You can now choose to allow these libraries to be excluded, in which case you can debug core files only. For more information, see Requiring Inclusion of Critical dbx Libraries in the Using the Debugging Window section of the Sun WorkShop online help.

Working With Projects

This release of Sun WorkShop uses projects to track the files, programs, and targets associated with your development projects and to build your programs without your needing to write a makefile. A project is a list that includes the files and the compiler, debugger, and build-related options used to build an executable, a static library or archive, a shared library, a Fortran application, a complex application, or a user makefile application.

In previous versions of Sun WorkShop, a workset was used instead of a project. Projects still have some workset features (there continue to be menu picklists that make it easy to access directories and files associated with a project), but a project includes more information about your program, such as which source files you want to build and how you want them built. If you have Sun WorkShop worksets, you can automatically convert your worksets to Sun WorkShop 6 projects when you load them. For more information, see Converting a Workset to a Project in the Working With Projects section of the online help.

When you start Sun WorkShop, the Welcome to Sun WorkShop dialog box opens and gives you immediate access to Sun WorkShop projects and the project wizard. Sun WorkShop also has project functions available from the Project Menu in the WorkShop Main Window to help you complete the following tasks:

- Create a new project or build a simple program using the project wizard and your own makefile or a makefile that Sun WorkShop creates for you (see Creating a New Project in the Working With Projects section of the online help)
- Change existing project settings, including how you want your project compiled and whether you want source browsing information generated (see Editing a Project and Edit Current Project Window in the Working With Projects section of the online help)

You can also choose to use the Sun WorkShop 6 programming environment without loading a project. Picklists keep track of the files, programs, directories, and targets associated with your development projects (see WorkShop Targets in the Building Programs section of the online help for more information). You can access each file, build the target, and debug the executable from the menus in the WorkShop Main Window. Build target information cannot be edited because it is not persistent; rather, it changes as you access, add, and remove build targets.

Sun WorkShop 6 TeamWare

TABLE 7 lists the new features available with the release of Sun WorkShop™ 6 TeamWare. Some of the features are described in greater detail in the sections following the table.

TABLE 7 Sun WorkShop 6 TeamWare

Feature	Description
Autofreezepointing	Autofreezepointing creates a freezepoint file for you before or after specific transactions. You can select the time that you want freezepoint files created: before or after bringovers, putbacks, undo actions, or resolve actions.
Versioning customized menu	The customized menu feature adds a new menu in the Versioning window titled "Customized," which provides access to your own commands. For information about creating a customized menu, see Creating a Customized Menu in the Managing Files section of the online help.
Delta comments	This new option adds delta comments to transaction output and email notification, including delta number, owner, and comments. In Configuring, choose Workspace ➤ Bringover Create/Bringover Update/Putback and check the Delta Comments box, or use the `-d` option with the bringover or putback command.
Merging diff navigator	The diff navigator appears between two unmerged files. You can click on the slide boxes on either side of the diff navigator to scroll through either file, or click on the arrows on the top or bottom to move the same distance in both files.
Configuring menu reorganization	Sun Workshop 6 TeamWare has implemented changes to the Configuring user interface.
Putback validation	When you turn on putback validation, only putbacks are allowed to the workspace. You can control which users can perform a putback and require that they have a specific password.
SCCS admin flags	Allows you to set SCCS admin flags for a file.
Workspace integrity check	New option to `workspace` command: `check [ -W ] [ - s ] wsname ...` checks files, access modes, parent-child relationships, and condition of the history file. The command exits with the following values: 0 = workspace is okay or 1 = error.

TABLE 7 Sun WorkShop 6 TeamWare *(Continued)*

Feature	Description
Workspace history viewer	Sun WorkShop TeamWare now includes an easy way to view the information in the workspace history file.
Workspace labels	With this feature, you can give a workspace a descriptive name that is more meaningful to your team. Choose Workspace ➤ Properties and select the Description tab.
New man page	`description`(4)

Configuring Menu Reorganization

Sun Workshop 6 TeamWare includes the following changes to the menus.

TABLE 8 TeamWare Menu Changes

Sun WorkShop TeamWare 2.2	Sun WorkShop 6 TeamWare
Configuring	
File ➤ Load Parent	Workspace ➤ Load Parent
File ➤ Load Children	Workspace ➤ Load Children
File ➤ Create Empty Child Workspace	Workspace ➤ Create Child
Edit ➤ Delete	Workspace ➤ Delete
Edit ➤ Rename	Workspace ➤ Rename
Edit ➤ Parent	Workspace ➤ Reparent
Edit ➤ Update ➤ Nametable	Workspace ➤ Update Nametable
Transactions ➤ Bringover ➤ Create	Actions ➤ Bringover Create
Transactions ➤ Bringover ➤ Update	Actions ➤ Bringover Update
Options ➤ Workspace	Workspace ➤ Properties
Options ➤ Workspace ➤ Edit Locks	Workspace ➤ Edit Locks
NEW	View ➤ Refresh
NEW	Workspace ➤ Properties ➤ Freezepointing

TABLE 8 TeamWare Menu Changes *(Continued)*

Sun WorkShop TeamWare 2.2	Sun WorkShop 6 TeamWare
NEW	Workspace ➤ Properties ➤ Putback Validation
NEW	Options ➤ Configuring ➤ Load Children: Selective/All
Versioning	
NEW	File ➤ File Info
NEW	Commands ➤ Uncheckout

Putback Validation

When you turn on putback validation, only putbacks are allowed to the workspace. The user is prompted for a password (Integration Request Identifier) before performing a putback. This feature only records the Integration Request Identifier, it does not check it. To check the Integration Request Identifier, you must write your own validation program. For more information, see Protecting Workspaces With Putback Validation in the Managing Workspaces section of the online help.

SCCS Admin Flags

You can set SCCS admin flags for a file by using Versioning, choose File ➤ File Info. For example, if you want Versioning to prompt for MRs (modification request strings) during putbacks, put the name of a validation program in the Validation Program box.

Workspace History Viewer

Sun WorkShop TeamWare now includes an easy way to view the information in the workspace history file (choose Workspace ➤ View History). In the Workspace History Viewer, you can view the transaction history of a workspace, transaction details, comments, and the command log. You can sort and filter the entries and search the comments and command log.

Sun WorkShop 6 Visual

TABLE 9 lists the new features available with the release of Sun WorkShop 6 Visual. Some of the features are described in greater detail in the sections following the table.

TABLE 9 Sun WorkShop 6 Visual

Feature	Description
Swing support	Sun WorkShop 6 Visual now has the ability to generate Java Swing code in addition to Java 1.0 and 1.1 code. As well as generating appropriate Swing components for the Motif widgets, Visual has increased the range of supported mappable resources: It is now possible to generate appropriate Swing code for toggles that contain images, shell icons, shell resize and delete response, the contents of lists, and rowcolumn entry alignment, among other extensions.
Enhanced windows support	Sun WorkShop Visual has added support for mapping X events to Windows MFC.
Integration with Sun WorkShop projects	Sun WorkShop Visual works with Sun WorkShop project wizards to help create projects with graphical user interfaces.

Swing Support

Sun WorkShop 6 Visual now has the ability to generate Java™ Swing code in addition to Java 1.0 and 1.1 code. As well as generating appropriate Swing components for the Motif widgets, Sun WorkShop 6 Visual has increased the range of supported mappable resources: It is now possible to generate appropriate Swing code for toggles that contain images, shell icons, shell resize and delete response, the contents of lists, and rowcolumn entry alignment, among other extensions.

The MWT class library, which maps Motif components into Java where the standard classes lack equivalence, has been ported to Swing in order to provide a more consistent look and feel. The dependence on the MWT to provide a Motif-compatible interface has been reduced as some of the Motif components are now mapped directly into an appropriate Swing component.

The Java Layout emulation widgets have been reworked where necessary to provide a more consistent behavior with respect to Java Layout characteristics.

As part of a move towards supporting cross-platform code for third- party (non-Motif) components, Visual now has the ability to specify default base classing for any integrated component. The classing can be on a general language basis or for specific variants. For example, it is possible to specify in a general way the proposed default class for Java, as well as Java 1.0, Java 1.1, and Swing-specific classing. Proposed component MFC classing can also be specified. The mechanisms address the problem wherein each given third-party component required individual manual configuration in order to create the right kind of object in the target language. Third-party specific resources are not mapped. The mechanisms do not address the following situations:

- A given component needs to be mapped into multiple native objects
- A compound component can be mapped by consideration of the constituent built-in parts.

Enhanced Windows Support

Sun WorkShop 6 Visual has added the following support for mapping X events to Windows MFC.

TABLE 10 Mappings for X Events to Windows MFC

MouseMotion	Generates a generic handler for any mouse movement with or without button press
ButtonPress	Generate all three handlers: Left, Center, and Right pressed handlers
ButtonRelease	Generate all three handlers: Left, Center, and Right release handlers
EnterWindow	`MouseActivate`
ExposureMask	`EraseBkgnd`
KeyPressMask	`WM_KEYDOWN`
KeyRelease	`WM_KEYUP`
KeymapstateMask	`WM_SYSKEYUP/WM_SYSKEYDOWN`
LeaveWindowMask	`WM_KILL_FOCUS`
ResizeRedirect	`WM_SIZE`
PropertyChangeMask	`ON_WM_PAINT`
VisibilityChangeMask	`WM_SHOWWINDOW`

Sun WorkShop 6 Performance Library

Sun Performance Library™ is a set of optimized, high-speed mathematical subroutines for solving linear algebra and other numerically intensive problems. Sun Performance Library is based on a collection of public domain applications available from Netlib at `http://www.netlib.org`. These routines have been enhanced and bundled as the Sun Performance Library.

TABLE 11 lists the new features available with the release of the Sun WorkShop 6 Performance Library. Some of the features are described in greater detail in the sections following the table.

TABLE 11 Sun WorkShop 6 Performance Library

Feature	Description
Support for LAPACK 3.0	LAPACK 3.0 subroutines have been added. The previous version of Sun Performance Library was based on LAPACK 2.0. The current version of Sun Performance Library still maintains compatibility with LAPACK 2.0 and LAPACK 1.x.
Sparse solver package	The sparse solver package provides routines for solving sparse matrices (symmetric, structurally symmetric, and unsymmetric coefficient matrices) using direct methods and a choice of fill-reducing ordering algorithms, including user specified orderings.
UltraSPARC III support	This release supports the UltraSPARC III instruction set architecture. To use the code that is specific to UltraSPARC III, compile with `-xarch=v8plusb` for 32-bit code or `-xarch=v9b` for 64-bit code.
Fortran 95 support	This release supports Fortran 95 language features. You can use the Sun Performance Library modules and definitions by including the `f95` statement `USE SUNPERF` in the code.
Changes to Sun Performance Library licensing	Sun Performance Library is no longer licensed. However, you should continue to use `-xlic_lib=sunperf` to ensure that the application links with the correct support libraries, and to ensure the correct version of Sun Performance Library is selected.

Fortran 95 Language Feature Support

By including the `f95` statement `USE SUNPERF` in your application, you can use the Sun Performance Library modules and definitions with the following features:

- **Type independence.** In Fortran 77 routines, you must specify the type as part of the name. In Fortran 95, a routine for a specific data type can be determined by the data type of the arguments passed to the routine.
- **Compile-time checking.** In Fortran 77, it is generally impossible for the compiler to know what parameters should be passed to a particular routine. In Fortran 95, the `USE SUNPERF` statement allows the compiler to know what the number, type, size, and shape of each parameter to each Sun Performance Library routine should be. It can check your calls against the expected value and identify errors during compilation.
- **Optional parameters.** In Fortran 77, all parameters must be specified in order for all routines. Fortran 95 allows some parameters to be optional. In Sun Performance Library, all increment parameters (`INCX`, `INCY`, and so on), workspaces, leading dimensions (`LDA`, `LDB`, and so on), and length or size parameters are optional.

For information on using these features and examples, see the *Sun Performance Library User's Guide*.

Changes to Sun Performance Library Licensing

Sun Performance Library is no longer licensed. However, you should continue to link using `-xlic_lib=sunperf` rather than `-lsunperf`. Use `-xlic_lib=sunperf` to ensure that the following occurs:

- The application is linked with the correct support libraries. In this release, Sun Performance Library is compiled with Fortran 95 instead of Fortran 77. Using `-xlic_lib` will correctly link in the Fortran 95 runtime libraries instead of the Fortran 77 runtime libraries.
- The correct version of Sun Performance Library is used. There are different versions of Sun Performance Library to support programs built with and without `-subparallel` and for different values of `-xarch`. Using `-xlic_lib=sunperf` will cause the driver to use the version of Sun Performance Library that best matches your command line options.

Sun WorkShop 6 Analyzer

Sun WorkShop 6 Analyzer is a complete rewrite of the Analyzer that was provided with Sun WorkShop 5.0.

TABLE 12 lists the new features available with the release of the Sun WorkShop 6 Analyzer. Some of these features are described in greater detail in the sections following the table.

TABLE 12 Sun WorkShop 6 Analyzer

Feature	Description
Function List is primary display	The Function List is the primary display, and is displayed by default when the Analyzer is invoked.
Multiple metrics	The Function List displays multiple metrics at the same time, instead of requiring you to select one category at a time to view. The Function List can also display metrics as values or a percentage.
Summary Metrics window	A new Summary Metrics window, accessed from the View menu, displays all metrics recorded for a selected function, both as values and percentages. The contents of the Summary Metrics window are independent of what appears in the function list display.
Callers-Callees window	From the Function List, you can access a new Callers-Callees window that shows how metrics are attributed from the callees of a selected function and to the callers of that function.
Generate annotated source code	You can now generate annotated source code for a selected function and display the results in an edit window.
Generate annotated disassembly	You can generate annotated disassembly for the selected function and display the results in an edit window.
Filter data by samples, threads, LWPs	You can now use the Select Filters dialog box to filter data by samples, threads, LWPs, or any combination of these. All displays and windows are updated to show data from the selected subset only.
Thread synchronization delay metrics	Two thread synchronization delay metrics are now available: a count of synchronization events exceeding the designated threshold, and the aggregate delay from those events.
Load multiple experiments	You can now load multiple experiments into the Analyzer at the same time. Their combined metrics appear in the Function List display.

Function List

The Function List is the primary display, and is displayed by default when the Analyzer is invoked.

Because the Function List can display multiple types of metrics at the same time, the Data list option menu has been redesigned to change only the display type. The Display list option menu has been removed.

You can now show metrics in the Function List as absolute values in seconds or counts, or a percentage of the total program metric, or both. By invoking a Select Metrics dialog box from the Function List display, you can do any of the following:

- Select the metrics displayed in the Function List
- Display metrics as counts, percentages, or both
- Specify which metric is used to sort the Function List
- Reorder the list

Callers-Callees Window

From the Function List, you can access a new Callers-Callees window that shows how metrics are attributed from the callees of a function and to the callers of that function. The Caller-Callees window shows the selected function in the center of the display, with callers of that function in the panel above and callees of the function in the panel below. For the selected function, the attributed metric represents usage within the function itself. For the callers above, it represents usage within the selected function and all functions it calls, as attributed up the callstack to its callers. For the callees below, it represents the proportion of the callee's metric that is attributable to calls from the selected function.

You can navigate through the program's structure in the Callers-Callees window by clicking on a function in either the caller panel or the callee panel; the display recenters on the newly selected function.

Generate Annotated Source Code

You can now generate annotated source code for a selected function and display the results in an edit window. Source code is annotated with per-line metrics, using the same set of metrics as the Function List. The source code also contains compiler parallelization commentary and Fortran 95 copyin and copyout commentary interleaved with the source. Source code display requires compiling with `-g` and will work for optimized code (`-g` no longer disables optimizations and parallelization).

Generate Annotated Disassembly

You can now generate annotated disassembly for a selected function and display the results in an edit window. Disassembly is annotated with per-instruction metrics, using the same set of metrics as the Function Display. Disassembly also contains compiler commentary and interleaved source code.

Metrics

The following new metrics or changes to existing metrics are provided with Sun WorkShop 6 Analyzer:

- Execution profile data is now called *clock-based profiling*. Execution profile data always includes called-function times.
- Histogram data is now called *exclusive metrics*.
- Cumulative data metrics are now called *inclusive metrics*.
- Clock-based profile data generates the following metrics:
  - Total LWP time
  - User CPU time
  - System CPU time
  - System wait time
  - Text-page fault time
  - Data-page fault time
  - Wall-clock time
- Thread synchronization delay tracing, a new kind of data, is provided that generates the following metrics:
  - A count of synchronization events exceeding the designated threshold
  - The aggregate delay from those events.

  These metrics are not collected by default.

Additional Changes

The Sun WorkShop 6 Analyzer provides the following additional changes:

- You can no longer select a subset of samples by clicking on them in the Overview display. The Select Samples text box and arrow buttons are no longer part of the Analyzer, and selection commands no longer appear in the View menu. You must do all filter selection of samples, threads, and LWPs in the Select Filters dialog box.

- You cannot delete an experiment record from inside the Analyzer. The Experiment ➤ Delete command has been changed to Experiment ➤ Drop. Use this command to drop an experiment from the Analyzer. The experiment record remains on disk until you remove it with the `er_rm` command.
- The format of `er_export` ASCII data has been changed. (`er_export` is used only for debugging.)

Sun WorkShop 6 Installation

TABLE 13 lists the new installation features available with the release of Sun WorkShop 6.

TABLE 13 Sun WorkShop 6 Installation

Feature	**Description**
Web Start	Web Start is the new installation software.
FLEXlm 7.0b	FLEXlm 7.0b is the license manager software provided with the Sun WorkShop 6 programming environment.

Sun WorkShop 6 Manuals, Man Pages, READMEs, and Online Help in HTML

The manuals, man pages, and READMEs in Sun WorkShop and Sun WorkShop TeamWare are available in HTML as well as in text files. The online help is now in HTML. You can access this HTML documentation at *install_directory*`/SUNWspro/docs/index.html`. In a default installation, *install_directory* is `/opt`.

Note – The information in this section assumes that your Sun WorkShop 6 products were installed in the `/opt` directory. Contact your system administrator if your Sun WorkShop software is not installed in `/opt`.

To view Sun WorkShop documentation that is available in HTML format, you must use Netscape™ Communicator 4.0 or a compatible Netscape version. Netscape Communicator is included in the Solaris™ 7 Operating Environment and the Solaris™ 8 Operating Environment.

If you are running the Solaris 2.6 Operating Environment and you do not have Netscape Communicator 4.0 or a compatible version, you can download Netscape Communicator 4.7 (for the Solaris 2.5.1 Operating Environment). Netscape Communicator 4.7 for the Solaris 2.5.1 Operating Environment is compatible with the Solaris 2.6 Operating Environment.

To download Netscape Communicator 4.7, see the following Netscape Communications Corporation Web site:

```
http://www.netscape.com/download/index.html
```

Sun WorkShop online help (in HTML) requires that you have JavaScript™ enabled, which is a default setting in Netscape preferences.